Module 2 Stacks Queues

21. Code to PUSH Element in a stack using Array  
void push() { int item; if(Top == MAXSIZE - 1) { printf("nThe Stack Is Full"); exit(0); } else { printf("Enter the element to be inserted "); scanf("%d",&item); Top= Top+1; stack[Top] = item; } }

22. Code to POP elements from stack using Array  
int pop() { int item; if(Top == -1) { printf("The stack is Empty"); exit(0); } else { item = stack[Top]; Top = Top-1; } return(item); }

23. Code to traverse the stack using Array  
void traverse() { int i; if(Top == -1) { printf("The Stack is Empty"); exit(0); } else { for(i=Top;i>=0;i--) { printf("Traverse the element "); printf("%dn",stack[i]);  
} }  
}

62. Code to insert an element in a circular queue  
void insert(int item) { if ((front == 0 && rear == MAX-1) || (front == rear+1)) { cout<<"Queue Overflow \n"; return; } if (front == -1) { front = 0; rear = 0; } else { if (rear == MAX - 1) rear = 0; else rear = rear + 1; }  
cqueue\_arr[rear] = item ; }

63. Code to delete an element from a circular queue  
void del() { if (front == -1) { cout<<"Queue Underflow\n"; return ; } cout<<"Element deleted from queue is : "<<cqueue\_arr[front]<<endl; if (front == rear) { front = -1; rear = -1; } else { if (front == MAX - 1) front = 0; else front = front + 1; } }

64. Code to display Circular Queue  
void display() { int front\_pos = front, rear\_pos = rear; if (front == -1) { cout<<"Queue is empty\n"; return; } cout<<"Queue elements :\n"; if (front\_pos <= rear\_pos) { while (front\_pos <= rear\_pos) { cout<<cqueue\_arr[front\_pos]<<" "; front\_pos++; } } else { while (front\_pos <= MAX - 1) { cout<<cqueue\_arr[front\_pos]<<" "; front\_pos++; }  
front\_pos = 0; while (front\_pos <= rear\_pos) { cout<<cqueue\_arr[front\_pos]<<" "; front\_pos++; } } cout<<endl; }

65. Code to insert an element in a priority queue  
void insert(int item, int priority) { node \*tmp, \*q; tmp = new node; tmp->info = item; tmp->priority = priority; if (front == NULL || priority < front->priority) { tmp->link = front; front = tmp; } else { q = front; while (q->link != NULL && q->link->priority <= priority) q=q->link; tmp->link = q->link; q->link = tmp; } }

66. Code to delete an element from priority queue  
void del() { node \*tmp; if(front == NULL) cout<<"Queue Underflow\n"; else { tmp = front; cout<<"Deleted item is: "<<tmp->info<<endl; front = front->link; free(tmp); } }

67. Code to display priority queue  
void display() { node \*ptr;  
ptr = front; if (front == NULL) cout<<"Queue is empty\n"; else { cout<<"Queue is :\n"; cout<<"Priority Item\n"; while(ptr != NULL) { cout<<ptr->priority<<" "<<ptr->info<<endl; ptr = ptr->link; } } }